

Configuration Manual -
Online News Popularity Prediction using
LSTM and Bi-LSTM

M.Sc. Research Project
M.Sc. Data Analytics

Prasad Rudrappa Shivu
Student ID: X19213077

School of Computing
National College of Ireland

Supervisor: Prof. Jorge Basilio

National College of Ireland
Project Submission Sheet
School of Computing



5 Student Name:	Prasad Rudrappa Shivu
Student ID:	x19213077
Programme:	M.Sc. Data Analytics
Year:	2021
Module:	M.Sc. Research Project
Supervisor:	Prof. Jorge Basilio
Submission Due Date:	16th August 2021
Project Title:	Online News Popularity Prediction using LSTM and Bi-LSTM
Word Count:	647
Page Count:	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Prasad Rudrappa Shivu
Date:	16th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual - Online News Popularity Prediction using LSTM and Bi-LSTM

Prasad Rudrappa Shivu
x19213077

1 Introduction

The Configuration handbook is a report that provides a step-by-step manual for creation, setup, implementation, and deployment of project ‘Online News Popularity Prediction using LSTM and Bi-LSTM’ presented in technical report. The purpose of this record is to aid through each step to obtain the preferred output, which can be furnished within the technical record. The whole undertaking is built with a variety of libraries, hardware, and software combinations.

1.1 Project Overview

The project objective is to predict the popularity of online news articles. The dataset, articles published by mashable.com, for the experiments is extracted from UCI machine learning repository. Recurrent Neural Network models, LSTM and Bi-LSTM, are used in this classification problem. .

2 Pre-requisites

The following are the prerequisites: The software and hardware configurations are presented below. To train the model for such a large image datasets, the GPU (Graphics Processing Unit) is required.

2.1 Hardware Requirements

- Processor Required: Intel(R) Core(TM) i5-10210U CPU @ 2.50GHz 1.60GHz, 2112 Mhz, 4 Core(s), 8 Logical Processor(s)
- RAM: 8GB
- System Type: 64 bit Operating Systems
- ROM: 1TB HDD
- Operating System: Windows 10

2.2 Software Requirements

1. Python Libraries listed below.

```
# Libraries import
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from copy import copy
from time import sleep
from scipy.stats import norm, probplot
from sklearn.preprocessing import StandardScaler
from keras.preprocessing.text import Tokenizer
from keras.models import Sequential
from keras.layers import Dense, GRU
from keras.layers import Dropout
from keras.layers import LSTM
from keras.callbacks import ModelCheckpoint
from keras.utils import np_utils
from keras.layers import Embedding
from keras.preprocessing.sequence import pad_sequences
import string
from textblob import TextBlob
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
import nltk
from keras.callbacks import EarlyStopping
from glob import glob
import numpy as np
from tqdm import tqdm
from keras.layers import Bidirectional, GlobalMaxPool1D
from sklearn.preprocessing import StandardScaler, RobustScaler
import os, re, csv, math, codecs
from numpy import unique
from numpy import hstack
from numpy import vstack
from numpy import where
from matplotlib import pyplot
from sklearn.datasets import make_blobs
from keras.layers.core import Dense, Activation
from keras.layers import LSTM, GRU, SimpleRNN, Bidirectional
from keras.layers import *
from keras.callbacks import ReduceLRonPlateau, ModelCheckpoint, EarlyStopping
import matplotlib.pyplot as plt
from sklearn.metrics import *
import xgboost as xgb
import pickle
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from random import random
import psutil
from sklearn.model_selection import train_test_split
from keras.utils import *
from imblearn.metrics import sensitivity_specificity_support
from matplotlib import pyplot
from tqdm import tqdm
from sklearn import preprocessing
import gc
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.decomposition import PCA
import warnings
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.model_selection import train_test_split, GridSearchCV
from collections import Counter
from imblearn.over_sampling import SMOTE
from collections import Counter
from keras.datasets import mnist
from keras.layers import Input, Dense
from keras.models import Model
from keras import backend as K
from tensorflow.keras.utils import to_categorical
```

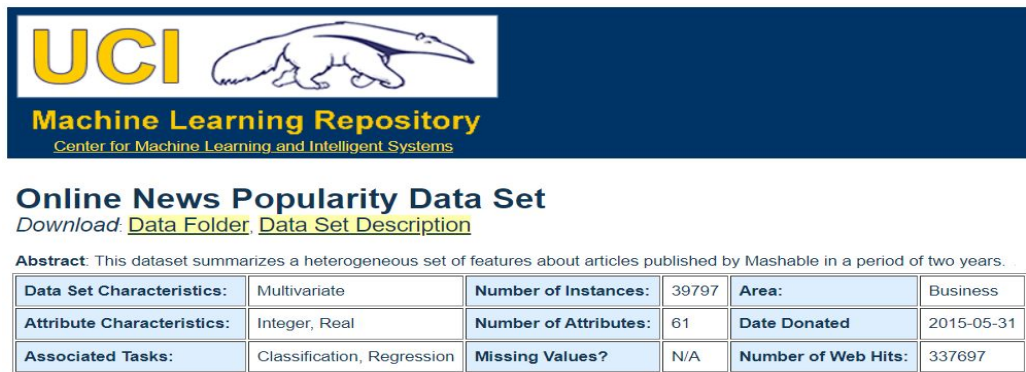
Figure 1: Python Libraries Used

2. Web Browser: The Google Chrome web browser version 87.0.4280.88 was used . Earlier versions of Google Chrome failed to support colab notebook.
3. Google Colaboratory (colab) IDE: This project is built with Google Colaboratory (colab) IDE, a free cloud service product by google allowing numerous Artificial Intelligence (AI) libraries, powerful GPU and TPU.
4. Google Colaboratory (colab) IDE: This project is built with Google Colaboratory (colab) IDE, a free cloud service that supports numerous Artificial Intelligence (AI) libraries.

3 Data Collection

The Datasets for this project is collected from UCI Machine Learning Repository. Steps for collecting the data are below:

- First the file is downloaded and stored in google drive [Figure 2](#)



UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Online News Popularity Data Set
Download: [Data Folder](#), [Data Set Description](#)

Abstract: This dataset summarizes a heterogeneous set of features about articles published by Mashable in a period of two years.

Data Set Characteristics:	Multivariate	Number of Instances:	39797	Area:	Business
Attribute Characteristics:	Integer, Real	Number of Attributes:	61	Date Donated	2015-05-31
Associated Tasks:	Classification, Regression	Missing Values?	N/A	Number of Web Hits:	337697

Figure 2: Dataset in UCI

- Google Drive is then mounted onto Google Colab, as the data for the experiment is stored on it. [Figure 3](#)

```
from google.colab import drive
drive.mount('/content/drive')
```

Go to this URL in a browser: <https://accounts.google.com/>

Enter your authorization code:
4/1AX4XfvjItmaAdpPhAvoC

Figure 3: Mounting Google Drive onto Google Colab

The dataset, containing the list of articles published was published in 2 years, by mashable.com.

4 Data Preprocessing & Exploratory Data Analysis

4.1 Preprocessing

The data was already preprocessed by K. Fernandes et al. There are slight modifications to be done for making it ready for exploratory data analysis. Dropping 'url' 'timedelta' and using 'shares' (a space character at the left) as a space character is appended to the left in the column headers.

```
# Here we drop the two non-predictive (url and timedelta) attributes. They won't contribute anything
data.drop(labels=['url', 'timedelta'], axis = 1, inplace=True)
data.head(n=4)

# describing the data
data.describe()

# from the data, there is a need to normalize to consider any machine learning model.
# creating a grading criteria for the shares
share_data = data[' shares']
```

Figure 4: Data Pre-processing

4.2 Exploratory Data Analysis

Data is explored in depth to analyse the attributes, their statistics like distribution and correlation. Few of the visuals can be seen in [Figure 5](#) and [Figure 6](#)

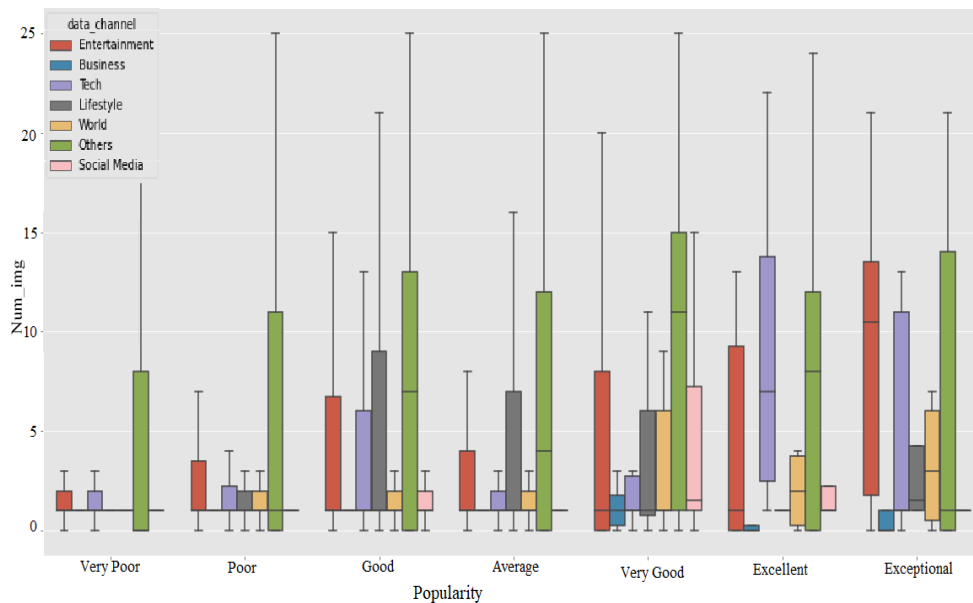


Figure 5: Data by News Channel

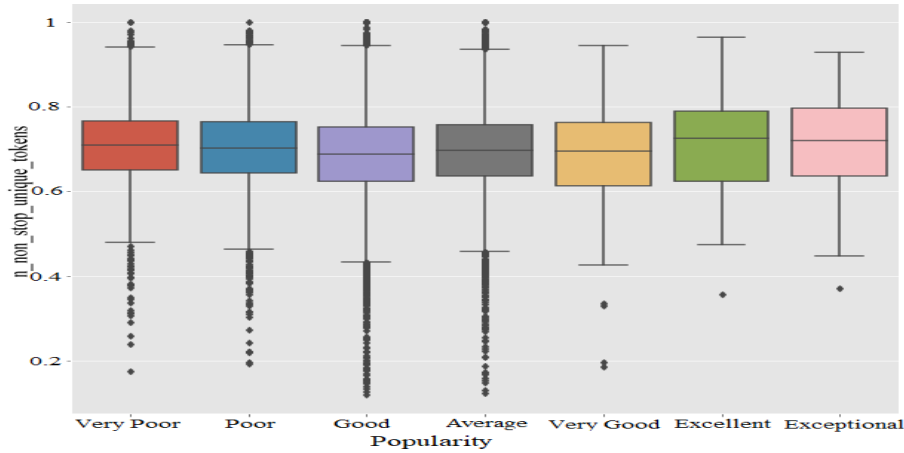


Figure 6: Popularity by n_non_stop_unique_tokens

5 Feature Selection

Six feature sets are used in the experiment - training and testing the model with LSTM and Bi-LSTM models.

```
# scaled all the feature selections
data_feature1_nor = scaler.fit_transform(data_feature1.values)
data_feature2_nor = scaler.fit_transform(data_feature2.values)
data_feature_fisher_nor = scaler.fit_transform(data_feature_fisher.values)
data_feature1_normal_nor = scaler.fit_transform(data_feature1_normal.values)
data_feature_fisher_normal_nor = scaler.fit_transform(data_feature_fisher_normal.values)
data_feature2_normal_nor = scaler.fit_transform(data_feature2_normal.values)

features_selection = [data_feature1_nor, data_feature2_nor, data_feature_fisher_nor, data_feature1_normal_nor,
                     data_feature_fisher_normal_nor, data_feature2_normal_nor]

features_selection_labels = ['Features on Hypothesis', 'All Features', 'Fisher based Features',
                            'Features on Hypothesis - Normal Distribution', 'Fisher based Features - Normal Distribution',
                            'All Features - Normal Distribution']
```

Figure 7: Feature Selection

6 Model Implementation

The models employed in predicting the news popularity are LSTM and Bi-LSTM, with and without Autoencoder.

The steps are as follows:

- Split the test and train data in the ratio 70/30 - [Figure 8](#)
- Deal with class imbalance by using the oversampling, SMOTE technique - [Figure 9](#)
- Reshape the dataset as required - [Figure 10](#)
- Apply the model


```
[ ] # Splitting the data for Training and Testing
# train and test for a feature selections
X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(data_feature1_nor, encoded_labels, test_size=0.3, shuffle=False)
X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(data_feature2_nor, encoded_labels, test_size=0.3, shuffle=False)
X_train_3, X_test_3, y_train_3, y_test_3 = train_test_split(data_feature_fisher_nor, encoded_labels, test_size=0.3, shuffle=False)
X_train_4, X_test_4, y_train_4, y_test_4 = train_test_split(data_feature1_normal_nor, encoded_labels, test_size=0.3, shuffle=False)
X_train_5, X_test_5, y_train_5, y_test_5 = train_test_split(data_feature_fisher_normal_nor, encoded_labels, test_size=0.3, shuffle=False)
X_train_6, X_test_6, y_train_6, y_test_6 = train_test_split(data_feature2_normal_nor, encoded_labels, test_size=0.3, shuffle=False)
```

Figure 8: Train-Test Data Split

```
[ ] print("WORKING OF SMOTE ALGORITHM")
print(" ")
print("Y_TRAIN Number transactions X_train dataset: ", data_feature1_nor.shape)
print("Y_TRAIN Number transactions y_train dataset: ", encoded_labels.shape)
print("Before OverSampling, counts of label '0': {}".format(sum(encoded_labels == 0)))
print("Before OverSampling, counts of label '1': {}".format(sum(encoded_labels == 1)))
print("Before OverSampling, counts of label '2': {}".format(sum(encoded_labels == 2)))
print("Before OverSampling, counts of label '3': {}".format(sum(encoded_labels == 3)))
print("Before OverSampling, counts of label '4': {}".format(sum(encoded_labels == 4)))
print("Before OverSampling, counts of label '5': {}".format(sum(encoded_labels == 5)))
print("Before OverSampling, counts of label '6': {}".format(sum(encoded_labels == 6)))

# import SMOTE module from imblearn library

sm = SMOTE(random_state = 2)
data_feature1_nor1,encoded_labels1= sm.fit_sample(data_feature1_nor, encoded_labels.ravel())
data_feature2_nor2,encoded_labels2= sm.fit_sample(data_feature2_nor, encoded_labels.ravel())
data_feature_fisher_nor3,encoded_labels3= sm.fit_sample(data_feature_fisher_nor, encoded_labels.ravel())
data_feature1_normal_nor4,encoded_labels4= sm.fit_sample(data_feature1_normal_nor, encoded_labels.ravel())
data_feature_fisher_normal_nor5,encoded_labels5= sm.fit_sample(data_feature_fisher_normal_nor, encoded_labels.ravel())
data_feature2_normal_nor6,encoded_labels6= sm.fit_sample(data_feature2_normal_nor, encoded_labels.ravel())
```

Figure 9: Dealing Class Imbalance

```
[ ] #from keras.utils import to_categorical
X_train_1=np.reshape(X_train_1,(X_train_1.shape[0],X_train_1.shape[1],1))
X_test_1=np.reshape(X_test_1,(X_test_1.shape[0],X_test_1.shape[1],1))
Y_TRAIN1=to_categorical(y_train_1,num_classes=7)
Y_TEST1=to_categorical(y_test_1,num_classes=7)
```

Figure 10: Reshaping after Treating Class Imbalance

6.1 Model 1: LSTM

Applying LSTM Model for the dataset considering the six combinations of features

```
▶ earllystopping = EarlyStopping(monitor = 'loss', verbose = 1,patience = 8, mode = 'min')

def model_stack(X_train_1,X_test_1,Y_TRAIN,Y_TEST):
    #input_img= Input(shape=shape) #(32,)
    model = Sequential()

    model.add(LSTM(128,input_shape=(X_train_1.shape[1],X_train_1.shape[2]),return_sequences=True))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.25))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.20))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.25))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.20))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.25))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.20))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.25))
    model.add((LSTM(64)))
    model.add(Dropout(0.20))
    model.add(Dense(7, activation='softmax'))
    print(model.summary())

    # compile network
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    # fit network
    checkpoint = ModelCheckpoint("check.h5", monitor='loss', verbose=1, save_best_only=True, mode='min')
    callbacks_list = [checkpoint,earllystopping] #
    model.fit(X_train_1, Y_TRAIN,batch_size=64, epochs=40, verbose=1,callbacks=callbacks_list,validation_data=(X_test_1,Y_TEST))
    #model.save("BILSTM.h5")
```

Figure 11: LSTM model

6.2 Model 2: Bi - LSTM

Applying Bi-LSTM Model for the dataset considering the six combinations of features

```
▶ earllystopping = EarlyStopping(monitor = 'loss', verbose = 1,patience = 8, mode = 'min')

def model_stack(X_train_1,X_test_1,Y_TRAIN,Y_TEST):
    #input_img= Input(shape=shape) #(32,)
    model = Sequential()

    model.add(LSTM(128,input_shape=(X_train_1.shape[1],X_train_1.shape[2]),return_sequences=True))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.25))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.20))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.25))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.20))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.25))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.20))
    model.add((LSTM(64,return_sequences=True)))
    model.add(Dropout(0.25))
    model.add((LSTM(64)))
    model.add(Dropout(0.20))
    model.add(Dense(7, activation='softmax'))
    print(model.summary())

    # compile network
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    # fit network
    checkpoint = ModelCheckpoint("check.h5", monitor='loss', verbose=1, save_best_only=True, mode='min')
    callbacks_list = [checkpoint,earllystopping] #
    model.fit(X_train_1, Y_TRAIN,batch_size=64, epochs=40, verbose=1,callbacks=callbacks_list,validation_data=(X_test_1,Y_TEST))
    #model.save("BiLSTM.h5")
```

Figure 12: Bi-LSTM model

6.3 Model 3: Autoencoder LSTM

Applying Autoencoder LSTM Model for the dataset considering the six combinations of features

```
def model_stack(shape,si,X_train_1,X_test_1,Y_TRAIN,Y_TEST):

    input_img= Input(shape=shape) #(32,)

    encoded = Dense(units=si, activation='relu')(input_img)
    encoded = Dense(units=100, activation='relu')(encoded)
    encoded = Dense(units=50, activation='relu')(encoded)
    encoded = Dense(units=25, activation='relu')(encoded)
    encoded = Dense(units=12, activation='relu')(encoded)
    encoded = Dense(units=6, activation='relu')(encoded)

    decoded = Dense(units=12, activation='relu')(encoded)
    decoded = Dense(units=25, activation='relu')(decoded)
    decoded = Dense(units=50, activation='relu')(decoded)
    decoded = Dense(units=100, activation='relu')(decoded)
    decoded = Dense(units=200, activation='relu')(decoded)
    decoded = Dense(units=si, activation='softmax')(decoded)

    autoencoder=Model(input_img, decoded)

    #We can extract the encoder which takes input data and outputs the encoded data of dimension 32
    encoder = Model(input_img, encoded)

    autoencoder.summary()

    encoder.summary()

    autoencoder.compile(optimizer='adam', loss='mse', metrics=['accuracy'])
```

Figure 13: Encoder LSTM model

6.4 Model 4: Autoencoder Bi - LSTM

Applying Autoencoder Bi-LSTM Model for the dataset considering the six combinations of features

```
def model_stack(shape,si,X_train_1,X_test_1,Y_TRAIN,Y_TEST):

    input_img= Input(shape=shape) #(32,)

    encoded = Dense(units=si, activation='relu')(input_img)
    encoded = Dense(units=100, activation='relu')(encoded)
    encoded = Dense(units=50, activation='relu')(encoded)
    encoded = Dense(units=25, activation='relu')(encoded)
    encoded = Dense(units=12, activation='relu')(encoded)
    encoded = Dense(units=6, activation='relu')(encoded)

    decoded = Dense(units=12, activation='relu')(encoded)
    decoded = Dense(units=25, activation='relu')(decoded)
    decoded = Dense(units=50, activation='relu')(decoded)
    decoded = Dense(units=100, activation='relu')(decoded)
    decoded = Dense(units=200, activation='relu')(decoded)
    decoded = Dense(units=si, activation='softmax')(decoded)

    autoencoder=Model(input_img, decoded)

    #We can extract the encoder which takes input as the input and the output of encoder is the encoded of dimension 32
    encoder = Model(input_img, encoded)

    autoencoder.summary()

    encoder.summary()

    autoencoder.compile(optimizer='adam', loss='mse', metrics=['accuracy'])
    #We finally train the autoencoder using the training data with 50 epochs and batch size of 256.
    autoencoder.fit(X_train_1, X_train_1,
                    epochs=120,
                    batch_size=64,
                    shuffle=True,
                    validation_data=(X_test_1, X_test_1))
    encoder = Model(input_img, encoded)
    encoder.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 14: Encoder Bi-LSTM model